Esercizi su prove formali di correttezza

università di ferrara
DA SEICENTO ANNI GUARDIAMO AVANTI.

Esercizio 1

- Dimostrare

{input1>0, input2>0}
read(z); read(y);
x:=1; j:=1;
while (j<=y) loop
  j:=j+1;
  x:=x*z
end loop
write(x);
{output = input1^input2}

Soluzione

- Backward substitution in Post:

x = input1^input2

- Detti
  - I = {x=z^(j-1) and j <= y+1}
  - cond = (j <= y) (condizione del ciclo while)
- I and not cond: j <= y+1 and j>y implica j=y+1, cioè x = z^y, quindi (se I è invariante e vale all'ingresso) poi basta dimostrare

  z^y = input1^input2

Soluzione

- I è invariante:
  - Con backward substitution,

  x*z = z^(j-1+1), cioè x = z^(j-1)
  - j + 1 <= y + 1, implicata da cond
- I vale all'ingresso: con backward substitution,
  - 1= z^0
  - 1 <= input2, implicata da Pre
- z^y = input1^input2: con backward substitution,

  input1^input2 = input1^input2

Esercizio 2

- Dimostrare:

```
{true}
function A (n: Integer) : Integer
var x, m : Integer
begin
    x := n; m := n;
    if m < 0 then
        while m < 0 do
        begin
                x := x + 2;
                m := m + 1;
        endwhile
    endif
    return (x)
end
{A = |n|}
```


Soluzione

- Con backward substitution: x = |n|
- Chiamiamo
  - IC l'istruzione condizionale
  - CW il ciclo while
- Se dimostriamo che
  - {m = n and x = n} IC {x = |n|}
- allora la prova si completa banalmente :
  - {true} x := n; m := n; {m = n and x = n}
- Dimostriamo che {m = n and x = n} IC {x = |n|}
  - {m = n and x = n and m < 0} CW {x = |n|}
  - {m = n and x = n and m >= 0} {x = |n|} (banale)

{m = n and x = n and m < 0} W {x = |n|}

- I = (x = 2m – n and m <= 0), cond = (m < 0)
- I è invariante: con backward substitution,
  - x + 2 = 2m + 2 – n (implicata da I)
  - m + 1 <= 0 (implicata da cond)
- All'uscita vale I and not cond:
  - (x = 2m – n and m <= 0) and m >= 0, quindi m = 0 e x = -n = |n|, poiché all'ingresso m = n e m < 0, quindi n < 0.
- All'ingresso I vale:
  - sostituendo, n = 2n – n
  - m <= 0 segue da n < 0

Scelta dell'invariante

- x (risp. m) parte da n e viene incrementato di 2 (risp. 1) a ogni iterazione, quindi

  x – n = 2 (m – n), cioè x = 2m – n
- Poiché vogliamo dimostrare che all'uscita x = -n, aggiungiamo una condizione tale che insieme a not cond implichi m = 0: not cond è m >= 0, quindi proviamo con m <= 0.